Computer Science • 13 (3) 2012 http://dx.doi.org/10.7494/csci.2012.13.3.23

Marcus Hilbrich
Ralph Müller-Pfefferkorn

IDENTIFYING LIMITS OF SCALABILITY IN DISTRIBUTED, HETEROGENEOUS, LAYER BASED MONITORING CONCEPTS LIKE SLAte

Abstract *In this paper we present the concept of a scalable job centric monitoring infrastructure. The overall performance of this distributed, layer based architecture called SLAte can be increased by installing additional servers to adapt to the demands of the monitored resources and users. Another important aspect is to offer a uniform global view on all data which are stored distributed to provide an easy access for users or visualisation tools. Additionally we discus the impact of these uniform access layer on scalability.*

Keywords distributed, scalable, SLAte, heterogeneous, grid, monitoring

1. Introduction

One of the consequences of using external, distributed resources for computing is to give up direct observability of computing tasks. The concept which admits this scientific challenge is called job centric monitoring. Job centric monitoring observes the jobs by recording monitoring data. With these data the job behaviour can be analysed. It allows to find reasons why a job aborted or to find unusual behaviour of jobs which are still running. Thus, it is possible to look into jobs instead of just knowing the current state or exit state of a job which is given by batch systems like PBS [1] [6] or grid middlewares like Globus [12] with its MDS information service. The state of the used computing resource are shown by resource monitoring systems like D-Mon [8], CMS Dashboard [7] or Ganga [20] but they do not give detailed information about specific jobs. The data about a job which is recorded by accounting systems like SGAS [11] or DGAS [17] gives only very basic information similar to batch systems. Thus analysing a job is not possible with these systems. As they produce just small amounts of data centralised architectures like a single database can be used. Job centric monitoring creates much more data. Thus, the demands of an architecture to transfer monitoring data are more challenging.

Profiling and tracing of applications (e.g. with Vampir [9], TAU [18] [18] or GNU gprof [3]) produces similar or even much larger amounts of data, but their major goal is to analyse a single job for application development or optimisation. Therefore, they only record data of an application which runs on a single HPC system. The collection of data from geographically distributed computing system is not covered by these systems.

One of the few job monitoring systems taking care of scalability issues is OCM-G [21]. It offers services to trigger actions based on external or monitoring events. To use it an application needs to be adapted and linked with the OCM-G libraries. Our solution has a more simple interface for recording monitoring data and there is no need to adapt the application or link any libraries.

The academic challenge of job centric monitoring is how to get from the old fashioned visualization concepts of tools like top [5] or the system monitor of the gnome desktop project [2], to a new kind of analysis and visualization systems like addressed by AMon [15] [16] [14]. Such a system has to offer methods to handle thousands of jobs running on different hardware with changing side effects like the influence of jobs of other users using the same computing systems.

But before we can launch into this challenge we have to develop methods to handle huge amounts of monitoring data generated in a distributed infrastructure like a cloud or a grid. If we consider a moderate system which is able to run 20 000 jobs at once we get 5 MB/s of monitoring data or 333 packages of 15 kB each per

second (assuming a measurement every 60 s and 15 kB of data[1] per measurement which is what we measured on current installations).

Job centric monitoring offers the most benefit on computing infrastructures which are able to run many jobs on resources which are shared by users. On these systems a user can not easily use desktop monitoring tools and the amount of jobs requires more sophisticated visualisation methods. These systems are e.g., a grid and a cloud as well as HPC systems. To handle the amount of data which is produced by such systems and to take into account the high number of small network packages we developed a concept for a layer based, distributed monitoring infrastructure. The implementation is called SLAte [13] which stands for **s**calable **l**ayered **a**rchi**te**cture. It allows to improve the performance to handle job centric monitoring data by installing additional servers.

This paper introduces the SLAte architecture and presents important implementation aspects and examples how the installation can be scaled. Based on the architecture, the system is analysed focusing on scalability. Potential bottlenecks are shown. This is followed by a outlook to future work and by conclusions.

2. SLAte architecture

The main idea of SLAte is to scale with the needs of the user community. If more users want to access monitoring data or computing systems are added it is possible to install additional servers to handle the additional load. Therefore the overall network bandwidth can be increased too. This is possible because each server installed on a new location increases the amount of network connections which are used by SLAte. For easy access to all data stored distributed over the servers of the infrastructure it was necessary to provide a unified and global view.

To get the idea of a scalable monitoring infrastructure to reality, a layer based concept was realised. A schematic view is shown in Fig. 1. Used are three layers. The Short Time Storage (STS) layer gets the data from the compute nodes and stores them temporarily. The Long Time Storage (LTS) layer accumulates the data from the STS servers and stores the data persistently and distributed over multiple servers. The outer layer is the Meta Data Service (MDS) which provides the global view of the data. In the following these layers are described in detail.

The STS servers should be installed locally at a site[2] (e.g., at the frontend of a grid computing resource) to be close to the computing nodes on which the monitoring data are produced. This is needed because a local network and a server within a site or a computing resource are able to handle many small network packages coming from the compute nodes much better than a wide area network. The small size of packages is caused by the decision to transfer each single measurement individually. In addition,

[1] Recorded monitoring data are e.g., user name, grid VO, timestamp, CPU time, load average, used and free memory, used and free swap space, IRQ load, job ID and host name.

[2] A site is a computing centre of one resource operator.

a low network transfer rate is is caused by other reasons like only partially filled network-(jumbo)-frames and the establishing of tcp and ssl connections. Another advantage of the transfer of the monitoring data in small packages is that if a job is aborted the monitoring data are not lost. It also avoids that the monitoring data use too many resources on the computing node (e.g., storage) which could block the application. It also prevents a transfer of all data at the end of the job. Thus, a new job can be started immediately.

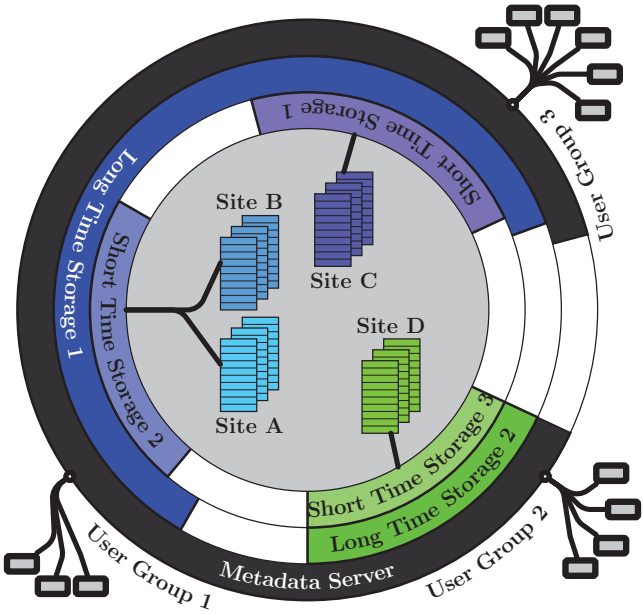


Figure 1. The layer based SLAte infrastructure for job centric monitoring data. Each layer can consist of multiple, locally distributed servers.

The monitoring data buffered on the STS server have to be moved to the LTS server. The connection between these two components is usually a wide and slow network where huge numbers of small packages can lead to problems. To avoid the network slowdown the data is repacked so that all measurements of one job are transferred as one object. To realise online monitoring it is also needed to get monitoring data of still running jobs. In this case the data on the LTS server are accessed and the data recorded afterwards have to be transferred in an additional package.

As like the STS layer, the LTS layer can consist of multiple servers. It is designed to store the monitoring data permanently in a distributed way. To join the monitoring data of multiple STS servers, one LTS server can accumulate the data delivered by several servers (in Fig. 1 this is the case for sites A, B and C).

Another task of the LTS servers is to make the data accessible to users and visualisation systems. Therefore each LTS offers services to search for job monitoring data and to access them.

To provide a unified view to the monitoring data distributed over the LTS servers the MDS layer was added (see Fig. 1). For scalebillity reasons this layer can also consist of multiple servers. An MDS server has a global view to all data of the more inner layers. Therefore the metadata (which includes the storage location) of all monitored jobs are stored on all MDS servers. The monitoring data itself are not stored on the MDS layer. A user or tool interested in monitoring data (like the visualization components of AMon) of a job has to know one of the MDS servers. Therefore a search request to an MDS server is answered with a list containing the location of all matching job monitoring data regardless on which LTS server the data is stored.

3. Implementation details

The implementation of the servers creating the three layers was done using the Grid middleware Globus (GT4) with its WSRF framework. The components are implemented as GT4 resources and services which communicate with each other and the client applications via their SOAP based web services. So the components can be deployed to already running GT4 servers (like the frontend of computing resources) or can be installed on separate hardware.

To handle data in a secure way the authentication and authorization mechanisms of GT4 are used. The STS, LTS and MDS use grid server certificates to authorize the communication. The authorization of the user is based on user certificates.

4. Exemplification of scalability

Based on the infrastructure of SLAte shown above we present an example how the infrastructure can be scaled. We consider different usage scenarios with rising load and explain how additional servers are used to handle the new demands.

Small installations can be build without an MDS server. If only one LTS server is installed it contains all metadata and the MDS server is not needed. Nevertheless, we are using an MDS sever for future expandability without changing the access point for users.

SLAte can be used by a single user or by a particular site. This is shown by “expansion stage 1” (one user observes jobs on one computing resource) and “expansion stage 2” (some resources on one site and several users) in Fig. 2(a). Such an installation can be done on one hardware server where STS, LTS and MDS services are deployed to a globus container.

“Expansion stage 3” and “4” (shown in Fig. 2(b)) represent installations to enable job centric monitoring on multiple sites. In such installations an STS server is needed on each site. These servers repack the small monitoring packages produced

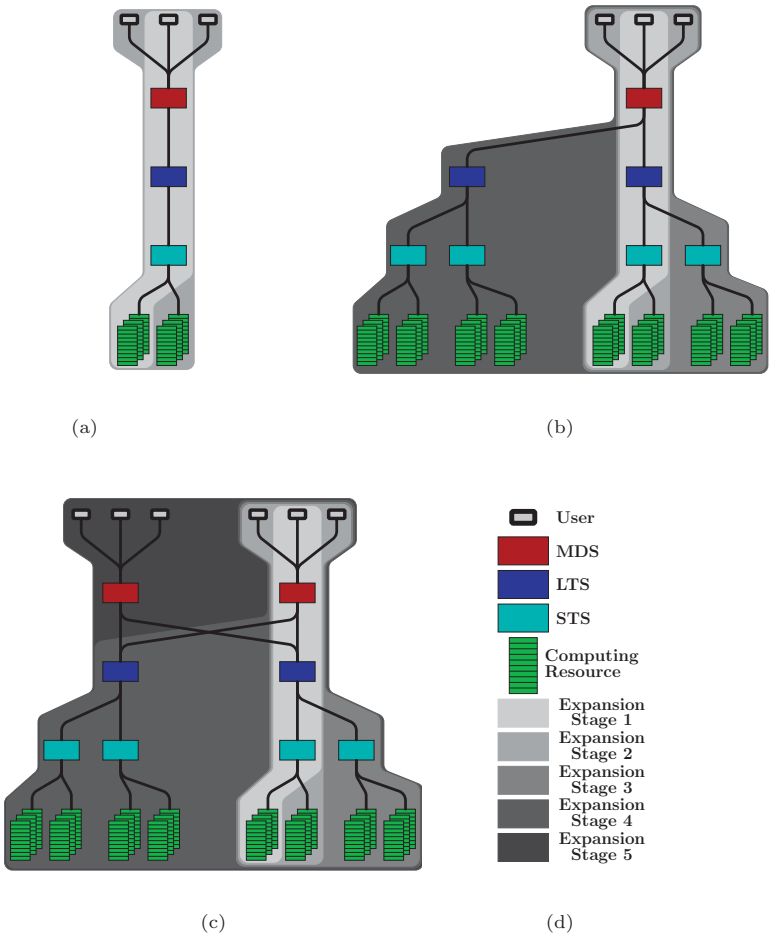


Figure 2. Example of different expansion states of a SLAte installation to handle a different amount of users and resources to monitor.

by each measurement on the computing resources to larger packages. This prevents a slowdown of the network between an STS and a LTS server.

If the persistent storage should be done in a distributed way which is useful to share the load of storage capacity and network it is needed to install additional LTS servers (shown by "expansion stage 4" in Fig. 2(b)). In such a case an MDS server is needed to get an uniform view to the data distributed over the LTS servers.

As like in the STS and LTS layer it is possible to share the load between several MDS servers. This is demonstrated by "expansion stage 5" in Fig. 2(c). In this installation multiple user groups are established where each one uses a separate MDS server.

The five expansion stages are just examples which show how scalability can be reached with additional servers. The number of components can be increased even further for larger computing infrastructures.

5. Scalability aspects

If an aspect of the monitoring architecture can be parallelised by adding an additional server without overhead the aspect is considered as scalable. In the following we show which tasks can be scaled up. If the parallelisation tends to overhead the aspect is considered in the next section.

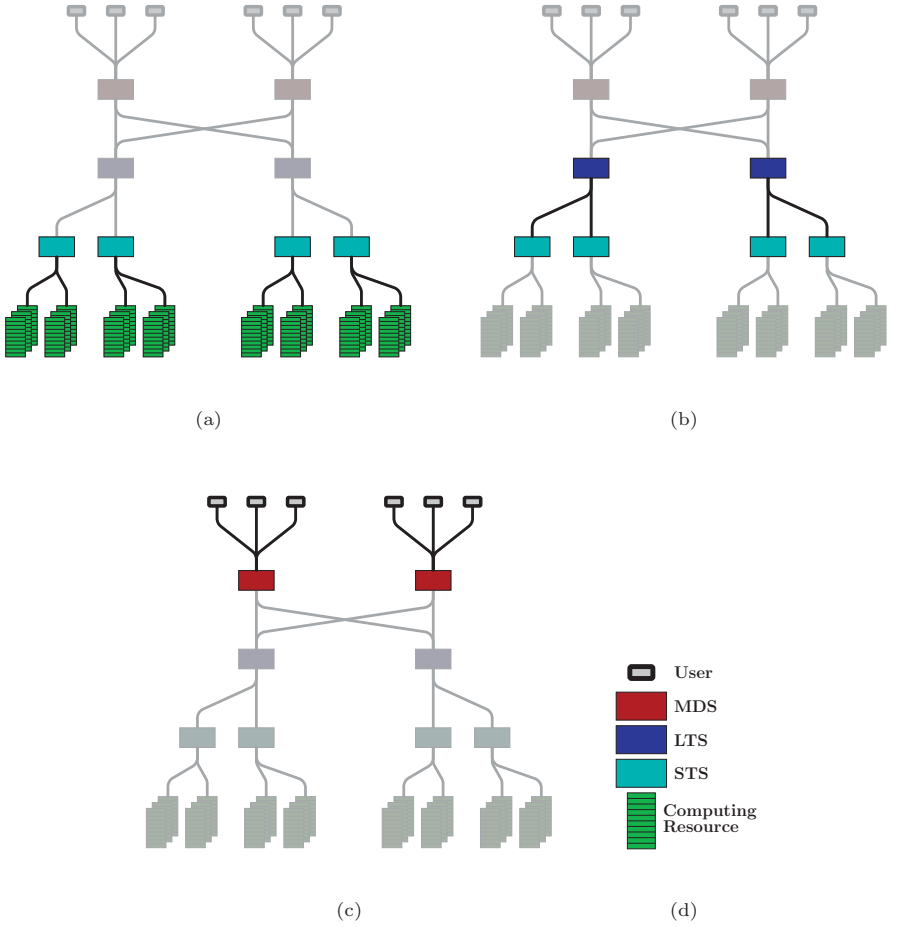


Figure 3. Shown is an example of a SLAte installation. The subfigures (a) to (c) highlight different scalable aspects of the installation.

The transfer of monitoring data from a computing node to an STS server (shown in Fig. 3(a)) is independent from transfers on another computing system with its own STS server. As conclusion the network connection from the nodes to the STS server has to handle the monitoring data of one computing system only. This has to be respected by sensors creating the monitoring data. In comparison with profiling and tracing (e.g, done by TAU [18], Supermon [19] or VNG [10]) the amount of monitoring data is quite small and can be easily handled by the network within a site.

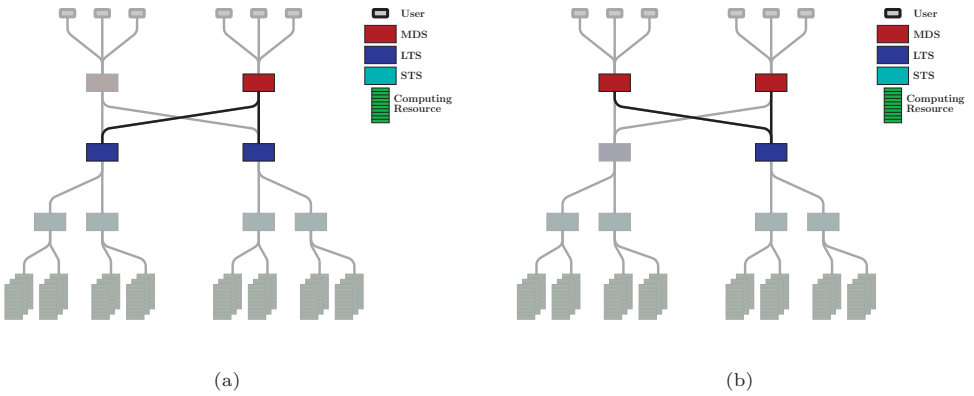
Storing, processing and sending the monitoring data to a LTS server are tasks which can be divided over multiple STS servers (visualised in Fig. 3(b)). Due to the fact that each job can be processed independent from other jobs these aspects are scalable.

LTS servers receive, process, and store monitoring data of jobs and allow users or visualisation systems to access them. For each job these tasks are independent of other job data and can be done on multiple servers without overhead. Another task is to search through the local data of a LTS server. If the resources of a server are not sufficient, the data can be distributed over multiple servers to lower the load.

The searching for monitoring data of jobs on the MDS layer (see Fig. 3(c)) can be distributed to multiple servers. This can be reached by using different servers for different user groups or by load balancing e.g. with pound [4].

6. Potential bottlenecks

Potential bottlenecks are aspects which can not be distributed over multiple servers at all or aspects which tend to overhead if they are done in a distributed way.



(a) (b)

Figure 4. Shown is an example of a SLAte installation. The subfigures highlight different potential bottlenecks of the installation.

Each MDS server has to keep all metadata of all jobs for search. Storing the metadata and transferring them to the MDS server (highlighted in Fig. 4(a)) is not scalable. But in comparison to the data the amount of metadata is very small[3]. Thus, the monitoring system can handle a much larger system.

Another aspect is that LTS servers have to transfer all metadata to all MDS servers (shown in Fig. 4(b)). This is an explicit overhead. To compensate this extra load it is possible to install additional LTS servers to lower the amount of data each server has to transfer. The load on the MDS servers which is analysed before is independent of the number of LTS servers.

7. Future work

To work out a concept of an widely scalable infrastructure is an interesting and challenging task. Such a concept has to be checked and potential problems have to be identified. The next task is to show that the system can be used in a real environment. This is ongoing work and the first results look promising.

We also started to develop a formal description of the SLAte architecture with its demands on network capacity and storage as well as its scalability. The results will be published soon.

An additional step is a prove of scalability under real circumstances and dedicated measurement environments. This work is in preparation.

8. Conclusion

The combination of a distributed storage system with central access points which provide a global view to all data requires to cumulate information about the stored monitoring data at multiple places. To allow to scale up to huge systems the information which has to be moved and stored to create this uniform view has to be as small as possible. This is realised by storing only metadata at the central access points.

All other aspects of managing job centric monitoring data with SLAte are scalable. This is realised by a concept which uses additional servers (added to the infrastructure of SLAte) to increase the overall performance and to fit the requirements of the users and computing systems.

Acknowledgements

Parts of the presented work are sponsored by Bundesministerium für Bildung und Forschung (grant number 01|G07014F).

[3]The metadata is about a tenth of a measurement. The monitoring data of a job can consists of hundreds of measurement depending on the measurement interval and the runtime of the job.

References

[1] A home for public OpenPBS patches, tools, and information. http://www.mcs.anl.gov/research/projects/openpbs/

[2] GNOME Documentation Library – System Monitor Manual V2.2. http://library.gnome.org/users/gnome-system-monitor/stable/index.html.en

[3] GNU gprof. http://sourceware.org/binutils/docs/gprof/

[4] pound. http://www.apsis.ch/pound/

[5] Procps – The /proc file system utilities. http://procps.sourceforge.net/index.html

[6] TORQUE Resource Manager. http://www.clusterresources.com/products/torque-resource-manager.php

[7] Andreeva J., Calloni M., Colling D., Fanzago F., D'Hondt J., Klem J., Maier G., Letts J., Maes J., Padhi S., Sarkar S., Spiga D., Mulders P. V., Villella I.: CMS Analysis Operations. *Journal of Physics: Conference Series*, 219(7):072007, 2010.

[8] Baur T., Breu R., Kálmán T., Lindinger T., Milbert A., Poghosyan G., Reiser H., Romberg M.: An Interoperable Grid Information System for Integrated Resource Monitoring Based on Virtual Organizations. *Journal of Grid Computing*, 7:319–333, 2009.

[9] Brunst H., Hackenberg D., Juckeland G., Rohling H.: Comprehensive Performance Tracking with Vampir 7. In Müller M. S., Resch M. M., Schulz A., Nagel W. E., editors, *Tools for High Performance Computing 2009*, pp. 17–29. Springer Berlin Heidelberg, 2010.

[10] Brunst H., Malony A., Shende S., Bell R.: Online Remote Trace Analysis of Parallel Applications on High-Performance Clusters. In Veidenbaum A., Joe K., Amano H., Aiso H., editors, *High Performance Computing*, volume 2858 of *Lecture Notes in Computer Science*, pp. 440–449. Springer Berlin / Heidelberg, 2003.

[11] Elmroth E., Gardfjäll P., Mulmo O., Åke Sandgren., Sandholm T.: A Coordinated Accounting Solution for SweGrid Version: Draft 0.1.3 Date: October 7, 2003.

[12] Foster I.: Globus Toolkit Version 4: Software for Service-Oriented Systems. In Jin H., Reed D., Jiang W., editors, *Network and Parallel Computing*, volume 3779 of *Lecture Notes in Computer Science*, pp. 2–13. Springer Berlin / Heidelberg, 2005.

[13] Hilbrich M., Müller-Pfefferkorn R.: A Scalable Infrastructure for Job-Centric Monitoring Data from Distributed Systems. In Bubak M., Turala M., Wiatr K., editors, *Proc. Cracow Grid Workshop '09*, pp. 120–125, ul. Nawojki 11, 30-950 Krakow 61, P.O. Box 386, Poland, February 2010. ACC CYFRONET AGH.

[14] Lorenz D., Borovac S., Buchholz P., Eichenhardt H., Harenberg T., Mättig P., Mechtel M., Müller-Pfefferkorn R., Neumann R., Reeves K., Uebing C., Walkowiak W., William T., Wismüller R.: Job monitoring and steering in D-Grid's

High Energy Physics Community Grid. *Future Gener. Comput. Syst.*, 25:308–314, March 2009.

[15] Müller-Pfefferkorn R., Neumann R., Hammad A., Harenberg T., Hüsken M., Mättig P., Mechtel M., Meder-Marouelli D., Borovac S., Ueberholz P.: Monitoring of Jobs and their Execution for the LHC Computing Grid. 2006.

[16] Müller-Pfefferkorn R., Neumann R., William T.: AMon – a User-Friendly Job Monitoring for the Grid. In Priol T., Vanneschi M., editors, *CoreGRID*, pp. 185–192. Springer, 2007.

[17] Piro Rosario M., Andrea G., Giuseppe P., Albert W.: Using historical accounting information to predict the resource usage of grid jobs. *Future Generation Computer Systems*, 25(5):499–510, 2009.

[18] Shende S. S., Malony A. D.: The Tau Parallel Performance System. *International Journal of High Performance Computing Applications*, 20(2):287–311, Summer 2006.

[19] Sottile M. J., Minnich R. G.: Supermon: A High-Speed Cluster Monitoring System. In *Proc. of IEEE Intl. Conference on Cluster Computing*, pp. 39–46, 2002.

[20] Vanderster D. C., Brochu F., Cowan G., Egede U., Elmsheuser J., Gaidoz B., Harrison K., Lee H. C., Liko D., Maier A., Mościcki J. T., Muraru A., Pajchel K., Reece W., Samset B., Slater M., Soroko A., Tan C. L., Williams M.: Ganga: User-friendly Grid job submission and management tool for LHC and beyond. *Journal of Physics: Conference Series*, 219(7):072022, 2010.

Affiliations

Marcus Hilbrich
Center for Information Services and High Performance Computing (ZIH), Technische Universität Dresden, `marcus.hilbrich@tu-dresden.de`

Ralph Müller-Pfefferkorn
Center for Information Services and High Performance Computing (ZIH), Technische Universität Dresden, `ralph.mueller-pfefferkorn@tu-dresden.de`

Received: 9.12.2011
Revised: 15.03.2012
Accepted: 9.07.2012